Proyecto de Software
Trabajo Hospital Dr. Alejandro Korn - Atención de pacientes en la guardia

En el presente ciclo lectivo trabajaremos en el desarrollo de un prototipo para el Hospital Dr. Alejandro Korn. La propuesta de trabajo surge de una serie de entrevistas realizadas por docentes de la cátedra al personal del servicio social del hospital. A lo largo de la cursada se desarrollará un prototipo funcional, que es un modelo a escala de la realidad (sin la funcionalidad total), el cual se irá enriqueciendo hasta llegar al producto final deseado.

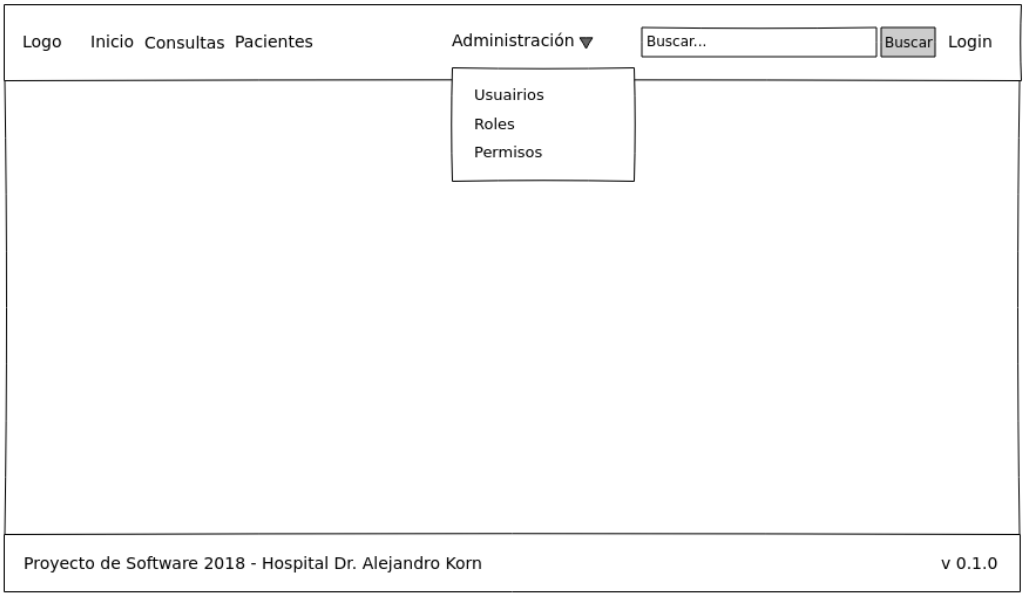
El trabajo se llevará a cabo en **tres etapas**, cada etapa finalizará con la entrega de un prototipo funcional que deberá presentarse en funcionamiento desde el servidor provisto por la cátedra.

Etapa 1 - v 0.1.0

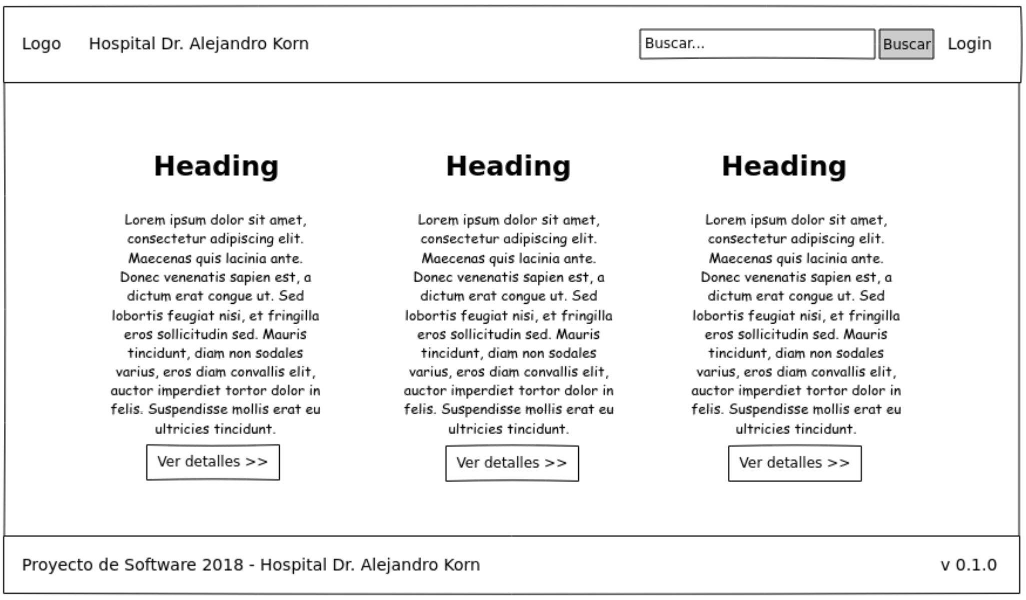
Fecha de entrega: 17/09 a las 8:00 (cierre del servidor)

En esta primera entrega se debe definir el **layout** y realizar 3 páginas en HTML5 y CSS3 que servirán como maqueta del prototipo a desarrollar a lo largo de la cursada. A continuación se presentan algunas pantallas a modo de ejemplo.

Layout: definirá una estructura general (división de áreas en la pantalla, que se mantendrá entre las distintas páginas) en donde se ubicarán los distintos componentes de la aplicación. El layout deberá estar compuesto al menos por: encabezado, pie de página, barra de navegación, menú y contenido.

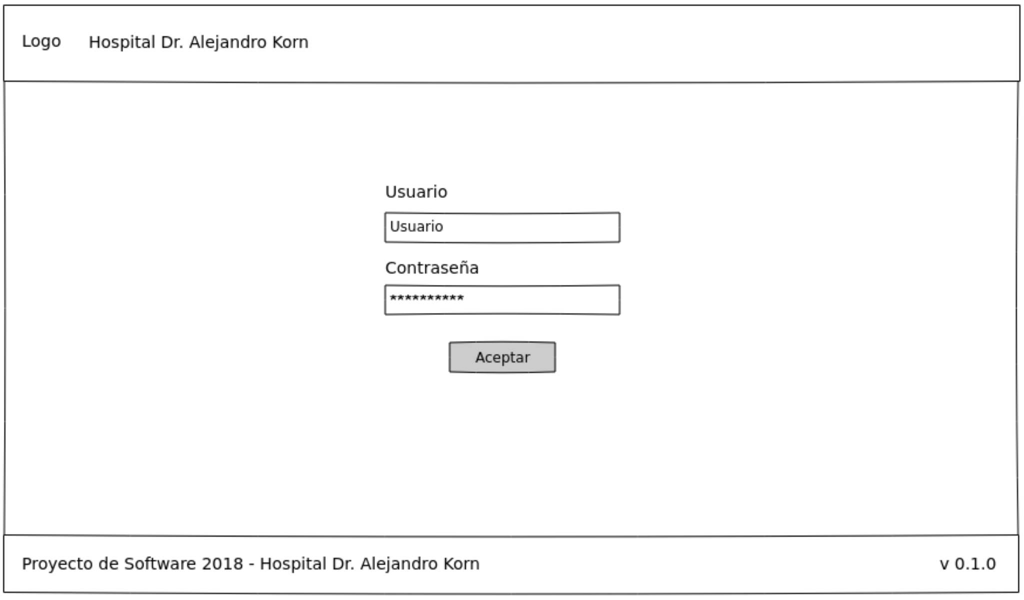


1.- Página de Inicio: contiene información pública del Hospital. Se debe poder acceder al login de alguna manera.

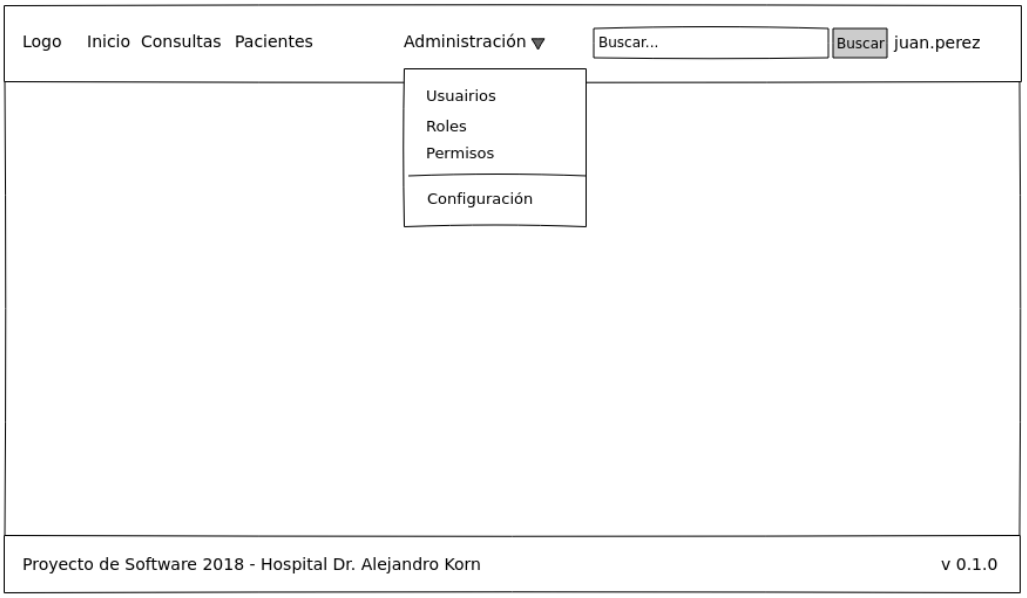


(El texto de los párrafos es ilustrativo, pueden usar otro o agregar más)

2.- Login de usuario: se debe presentar un formulario con un campo para el usuario y otro para la contraseña, además se deberá incluir un botón "Aceptar".
Los datos ingresados por el usuario **se deben valida**r mediante HTML5 (que no estén vacíos).



3.- Administración: se muestra el menú con las operaciones que puede realizar el usuario del sistema con los roles adecuados.



Realizar un informe en el que se detallen los errores de validación de CSS y HTML y que medidas tomaron para solucionarlos. La Cátedra pondrá a disposición una plantilla para el informe, el mismo deberá ser entregado a los/las ayudantes del grupo.

Consideraciones generales

- Como regla general tenga en cuenta que **TODAS las páginas** de cada entrega se deben **visualizar correctamente** tanto en un sistema Linux como Windows utilizando diversos navegadores y **deben pasar los validadores** provistos por la **W3C** tanto para HTML (http://validator.w3.org/) como para CSS (http://jigsaw.w3.org/css-validator/).
- **Todos y todas los/as integrantes del grupo deben realizar commits** de los archivos de la entrega, de lo contrario el/la alumno/a desaprobará la entrega.
- Las páginas deben **adaptarse a la pantalla** del visitante (**web responsive**) mediante directivas de estilos en los archivos .css. Considere al menos **un ancho mínimo de pantalla para ajustar los estilos. Es decir que apliquen ciertos estilos cuando el ancho de la pantalla es menor a (por ejemplo) 768px, y otros cuando es mayor.**
- Utilice el sistema Simor (http://simor.linti.unlp.edu.ar) para **evaluar la accesibilidad** de las páginas a entregar y corregir lo necesario para que cumpla con las principales pautas WCAG.
- Para esta etapa **NO** puede utilizarse **Bootstrap** ni otro framework similar.
- Los bocetos **NO** son orientativos, deben respetarse y a partir de ellos agregar lo necesario.
- Para esta etapa, la entrega es **obligatoria** (no su aprobación) y **los/as integrantes NO deben presentarse a la defensa**.
- **Importante**:

- El proyecto podrá ser realizado de modo individual o en grupos de dos o tres integrantes (será responsabilidad de los estudiantes la conformación de los equipos de trabajo). Todos los estudiantes cumplirán con la totalidad de la consigna, sin excepciones.
- La repetición de código podría implicar la desaprobación de la entrega

Etapa 2 - v 0.2.0

Fecha de entrega: 22/10 a las 8:00 (cierre del servidor)

Fecha de reentrega: 5/11 a las 8:00 (cierre del servidor)

2.1 Manejo de sesiones

Deberá implementarse un manejo de sesiones adecuado, verificando la sesión y roles cuando corresponda. Para cada módulo se indicará si requiere autenticación o no y roles necesarios.

2.2 Módulo de usuarios

Desarrollar el **módulo usuarios** que deberá contemplar al menos la siguiente funcionalidad:

- CRUD[1] de usuarios, validar la existencia de un usuario, es decir, no pueden existir dos usuarios con el mismo nombre de usuario.
- Se deben poder realizar búsquedas sobre los usuarios, al menos por los siguientes campos:
  - nombre de usuario.
  - activo/bloqueado.

  El resultado de la búsqueda debe estar paginado en base a la configuración del sistema (ver **módulo de configuración**).
- Activar/Bloquear usuario.
- Asignar o desasignar roles de un usuario (pueden ser varios).

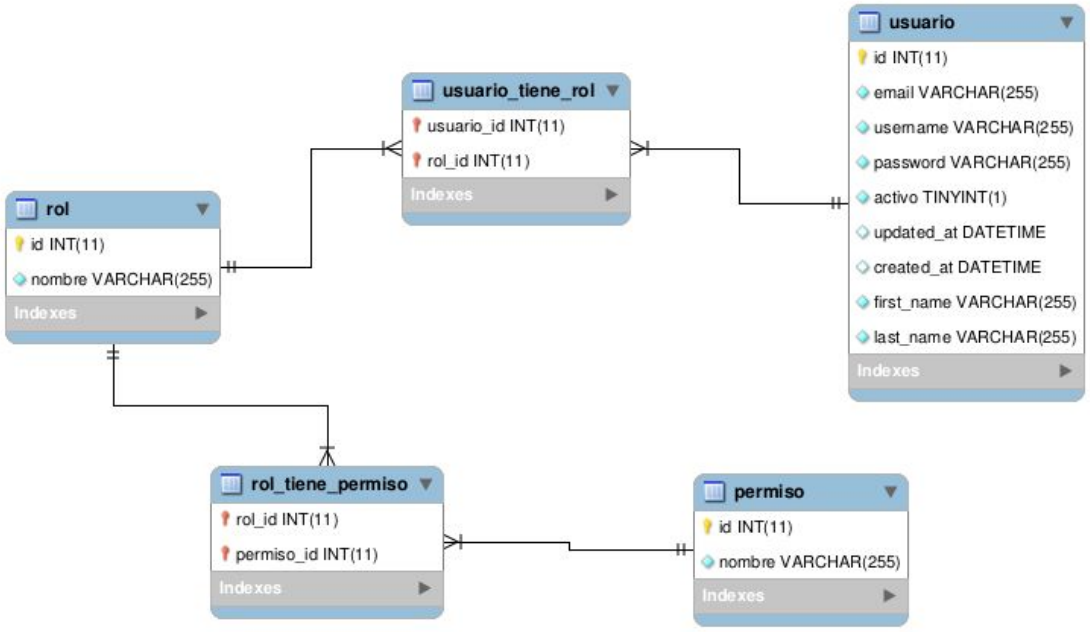
Para esta etapa, no será obligatorio desarrollar el CRUD de los roles y los permisos, podrán administrarse desde la base de datos. Los usuarios, roles y permisos sólo podrán ser administrados por un usuario con rol de **Administrador**.
Considerar que un usuario podrá tener más de un rol, y para cada rol se pueden configurar varios permisos. Los permisos necesarios asociados a cada rol deberán deducirse del enunciado, ante la duda **consulte a su ayudante**.

El nombre de los permisos deberá respetar el patrón **modulo_accion**, por ejemplo, el módulo de pacientes (ver **módulo de pacientes,** siguiente etapa) deberá contemplar los siguientes permisos:

[1] Create Read Update Delete: creación, lectura (listado), actualización y borrado de usuarios (ABM).

- paciente_index: permite acceder al index (listado) del módulo.
- paciente_new: permite cargar un paciente.
- paciente_destroy: permite borrar un paciente.
- paciente_update: permite actualizar un paciente.
- paciente_show: permite visualizar un paciente.

Modelo de Usuarios, roles y permisos



2.3 Módulo de configuración

Este módulo permitirá administrar la configuración del sistema, como mínimo deberá contemplar la siguiente configuración:

- Información del Hospital
  - Título
  - Descripción
  - Mail de contacto.
- Cantidad de elementos por página en los listados del sistema (todos los listados deberán respetar este valor para el paginado).
- Sitio habilitado/deshabilitado: si se selecciona esta opción, se debe deshabilitar el frontend, mostrando el siguiente texto “El sitio se encuentra mantenimiento”.

La configuración del sistema sólo podrá modificarla un usuario con el rol de **Administrador**.

2.4 Módulo de Pacientes

Una vez que el paciente llega al hospital es recibido/a en la Guardia, en ese momento personal del área verifica si el/la paciente se encuentra cargado en el sistema, para esto realiza una búsqueda por apellido, nombre, tipo y número de documento del/la paciente o por número de historia clínica[2]. De no poder obtener estos datos, se ingresa como "NN" y se le asigna un número de historia clínica. En caso de no encontrar al paciente, un usuario con rol de **EquipoDeGuardia** realiza la carga en el sistema.
Desarrollar el CRUD de **Pacientes** que permita registrar sus datos personales. En la Etapa 3, los datos de referencia (tipo de doc., partido, localidad y obra social) deberán obtenerse de la **API de Referencias** proporcionada por la cátedra[3]. En esta oportunidad se podrá fijar un valor seleccionado de una lista estática de elementos.
El alta (acción new) de pacientes sólo podrá realizarlo un usuario con el rol de **EquipoDeGuardia.**

A continuación se definen los roles necesarios para el resto de las acciones:

- index, show, update: **EquipoDeGuardia**
- destroy: **Administrador**

Los datos necesarios del paciente son:

- Apellido*: apellido del paciente (text)
- Nombre*: nombre del paciente (text)
- Fecha de nacimiento*: fecha de nacimiento del paciente (date)
- Lugar de nacimiento: lugar de nacimiento del paciente (text)
- Partido[4]: listado de partidos (select). Utilizar AJAX para cargar el listado de localidades
- Región Sanitaria[5]: región sanitaria a la que pertenece (texto sólo lectura). Dependiente del partido.
- Localidad[6]: listado de localidades dependiente del partido seleccionado (select)
- Domicilio*: domicilio del paciente (text)
- Género*: Masculino | Femenino | Otro (select)
- Tiene en su poder su documento *: indicar si el/la paciente tiene el documento en su poder. SÍ | NO (boolean)
- Tipo de doc.*[7]: tipo de documento del paciente (select)
- Número de documento*: número de documento del paciente (number)
- N° de historia clínica: número de hasta 6 cifras (number)
- N° de carpeta: número de hasta 5 cifras (number)

[2] Número de hasta 6 cifras.
[3] https://api-referencias.proyecto2018.linti.unlp.edu.ar/tipo-documento y https://api-referencias.proyecto2018.linti.unlp.edu.ar/obra-social
[4] en Etapa 3 se obtiene de la API de Referencias.
[5] en Etapa 3 se obtiene de la API de Referencias.
[6] en Etapa 3 se obtiene de la API de Referencias.
[7] en Etapa 3 se obtiene de la API de Referencias.

- Tel/Cel: teléfono o celular del paciente (text)
- Obra social[8]: listado de obras sociales (select)

Nota: los campos que tienen (*) son obligatorios.

Consideraciones generales

- El prototipo debe ser desarrollado utilizando PHP, HTML5, CSS3 y MySQL, y **respetando el modelo en capas MVC**. Debe utilizar PDO como mecanismo de abstracción de bases de datos, teniendo la posibilidad de utilizar Doctrine como ORM[9]. Debe tener en cuenta los conceptos de semántica web proporcionada por HTML5 siempre y cuando sea posible con una correcta utilización de las etiquetas del lenguaje.
- El trabajo será evaluado **desde el servidor de la cátedra** que cada grupo deberá gestionar mediante Git. **NO se aceptarán entregas que no estén realizadas en tiempo y forma en el servidor provisto por la cátedra**.
- Cada grupo tendrá un ayudante a cargo, quien evaluará el progreso y la participación de cada integrante mediante las consultas presenciales y el seguimiento mediante GitLab.
- Toda vista (**HTML5** y **CSS3**) debe validar contra las especificaciones de la W3C (http://validator.w3.org/ y https://jigsaw.w3.org/css-validator/ respectivamente). En esta oportunidad puede utilizar **Bootstrap** u otro framework similar para realizar el maquetado.
- **El uso de Twig como motor de plantillas es obligatorio**.
- **NO** puede utilizar un framework PHP para el desarrollo.
- Para esta etapa, la entrega y aprobación son obligatorias. Todos los integrantes deben presentarse a la defensa.
- Tener en cuenta que el servidor de la cátedra utiliza los siguientes servicios y versiones:
    - Servidor Web: Apache 2.4.29
    - Servidor de Base de Datos: MariaDB 10.1.34
    - Intérprete PHP: Php 7.2

- **Importante**:
    - El proyecto podrá ser realizado de modo individual o en grupos de dos o tres integrantes (será responsabilidad de los estudiantes la conformación de los equipos de trabajo). Todos los estudiantes cumplirán con la totalidad de la consigna, sin excepciones.
    - La repetición de código podría implicar la desaprobación de la entrega

[8] en Etapa 3 se obtiene de la API de Referencias.
[9] Object-Relational Mapping

Etapa 3 - v 0.3.0

Fecha de entrega: 19/11 a las 8:00 (cierre del servidor)

Fecha de reentrega: 3/12 a las 8:00 (cierre del servidor)

3.1 Módulo de Atención de Pacientes

Una vez que el personal del Área de Recepción realiza la carga de los datos del paciente, se continúa con los datos de la atención, indicando a donde debe ser derivado el paciente según su domicilio.

Se deberá desarrollar el CRUD del Módulo de Atención de Pacientes. Los datos de referencia de la derivación, deberán obtenerse de la **API de Instituciones** proporcionada por la cátedra[10].
Los datos de consulta y derivación sólo podrán cargarlos o modificarlos usuarios que dispongan del rol de **EquipoDeGuardia.**
Al momento de registrar la atención de un/una paciente, se deberá visualizar en un mapa las derivaciones previas que se le han realizado al mismo. Por lo tanto, se deberá guardar la historia de las derivaciones realizadas para cada paciente.
A continuación se definen los roles necesarios para el resto de las acciones:

- show, update: **EquipoDeGuardia**
- destroy: **Administrador**

Los datos necesarios para el módulo de atención son los siguientes:

- Paciente*: paciente cargado previamente (Autocomplete). Obtener mediante consulta AJAX.
- Fecha*: fecha en la que se realiza la consulta (date).
- Motivo*[11]: motivo de la consulta (select).
- Derivación[12]: institución donde ha sido derivado el/la paciente (select).
- Articulación c/otras Instituciones: por ej. “Hablé con Juan Carlos para coordinar la derivación del paciente XXX” (text)
- Internación*: indica si la atención derivó en una internación o no (boolean - SI | NO)
- Diagnóstico*: diagnóstico realizado al paciente (text)
- Observaciones: observaciones generales (text)
- Tratamiento farmacológico: lista fija, Mañana, Tarde o Noche (select)

[10] https://api-referencias.proyecto2018.linti.unlp.edu.ar/
[11] Utilizar el listado del dump de la base de datos facilitado por la Cátedra, no es necesario realizar el ABM.
[12] Acceso a API de Instituciones

- Acompañamiento: indica quién acompañó al paciente (select). El listado es el siguiente: Familiar Cercano; Hermanos e hijos; Pareja; Referentes vinculares; Policía; SAME; Por sus propios medios.

Nota: los campos que tienen (*) son obligatorios.

3.2 Acceso a API de Referencias

Para los módulos Pacientes y Atención de Pacientes, adaptar la carga obteniendo los datos de la API. La cátedra pone a disposición la **API de Referencias**, que podrá ser consultada para obtener los datos necesarios al momento de cargar un Paciente, y la atención.

Datos de referencia que proveerá la API:

- Obra Social
- Tipo de Documento
- Localidad
- Partido

API de Referencias
URL base: https://api-referencias.proyecto2018.linti.unlp.edu.ar
Autenticación: con token provisto por la cátedra.

En la siguiente dirección podrán consultar la documentación de la **API de Referencias**
https://inicio.proyecto2018.linti.unlp.edu.ar/api-referencias-doc/index.html

3.3. API de Instituciones

Desarrollar la **API de Instituciones** que permita consultar las Instituciones a partir de una región sanitaria o el identificador de la institución.
Una vez desarrollada la API, se deberá consultar utilizando el **Cliente de Telegram** o un **Cliente Web** desarrollado con Vue.js.

La API de Consulta de Instituciones tendrá al menos los siguientes endpoints[13]:

1) **GET:** /instituciones/
   Listado de Instituciones. Este servicio permite obtener el listado completo de las Instituciones disponibles.

2) **GET:** /instituciones/**{:institucion-id}**
   Obtener una Institución. Este servicio permite obtener una Institución según el ID pasado por parámetro.

3) **GET**: /instituciones/region-sanitaria/**{:region-sanitaria}**

[13] punto de acceso a un servicio para una Arquitectura Orientada a Servicios

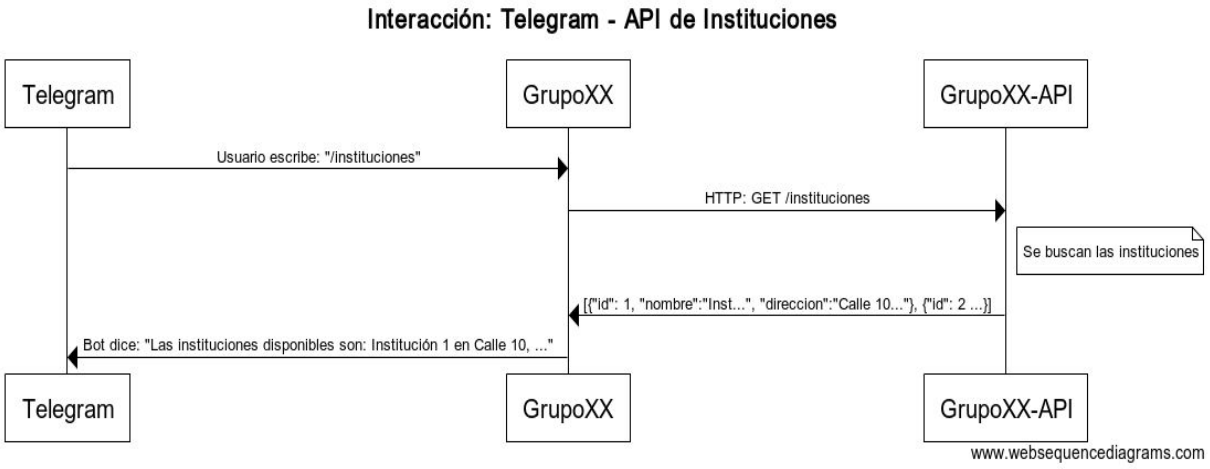
Listado de Instituciones. Este servicio permite obtener el listado completo de Instituciones de una regiones sanitaria a partir de un número de región pasado por parámetro.

Cliente Telegram de la API de Instituciones

Desarrollar un cliente PHP utilizando **Telegram** que permita interactuar con la **API de Instituciones** desarrollada en 3.3. Se deberá crear una bot de Telegram (https://core.telegram.org/bots) para consultar los servicios de la API. Deberá tener (al menos) 2 comandos:

● /instituciones: Devolverá un listado de Instituciones disponibles.
● /instituciones-region-sanitaria: **region-sanitaria**: Devolverá un listado de Instituciones a partir de una la región sanitaria indicada por parámetro.

Las consultas realizados desde el **Bot de Telegram** deben recibirse y procesarse realizando llamados a la **API de Instituciones**.



Nota: la API de Instituciones NO puede desarrollarse con un generador de código.

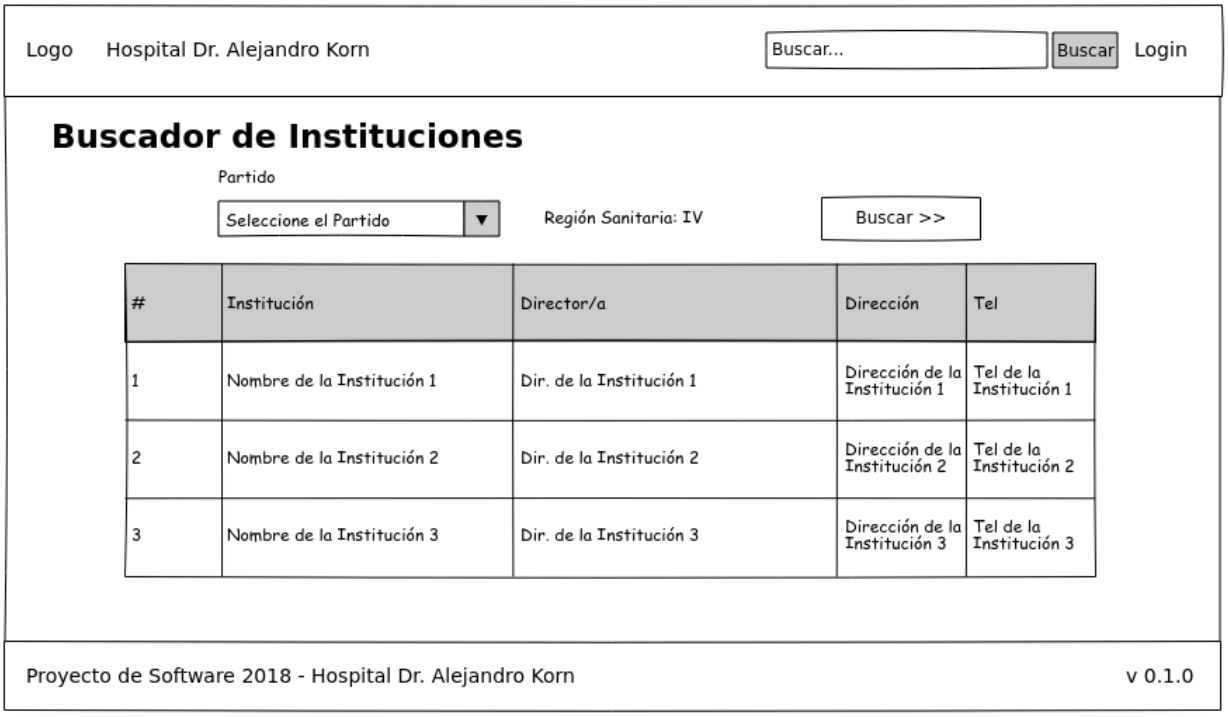
Cliente Web para la API de Instituciones

Desarrollar un Frontend Web utilizando Vue.js,[14] que permita interactuar con la **API de Instituciones** desarrollada en 3.3**.** El cliente deberá acceder a:

- **GET:** /partidos/ servicio necesario para armar el listado de partidos
- **GET:** /regiones-sanitarias/**{:region-sanitaria-id}**[15] servicio necesario para indicar el nombre de la región sanitaria del partido seleccionado
- **GET**: /instituciones/region-sanitaria/**{:region-sanitaria}** servicio necesario para listar las instituciones de la región sanitaria, a partir del partido seleccionado

[14] Framework JavaScript - https://vuejs.org
[15] Se obtiene de la API de Referencias de la Cátedra

El frontend debe desarrollarse con Vue.js, no será necesario autenticar el cliente web ya que se trata de una consulta pública.
Se deberá poder consultar a qué Institución debe asistir un ciudadano a partir del Partido, ingresado[16].
Una vez seleccionado el Partido se deberá indicar, en texto de solo lectura, la región sanitaria, y listar las instituciones que pertenecen a la misma. Tenga en cuenta que tanto la región sanitaria como las instituciones deberán consultarse vía servicio.



3.4 Reportes

Desarrollar el módulo de reportes que contemple los siguientes listados:

- Listado de consultas agrupadas por motivo. Realizar un gráfico de tortas. Exportar a PDF el listado y el gráfico.
- Listado de consultas agrupadas por género. Realizar un gráfico de tortas. Exportar a PDF el listado y el gráfico.
- Listado de consultas agrupadas por Localidad. Realizar un gráfico de tortas. Exportar a PDF el listado y el gráfico.

Para los reportes se puede utilizar cualquier librería siempre y cuando sea de uso libre, algunos ejemplos podrían ser: Highcharts (http://www.highcharts.com), Charts.js (http://www.chartjs.org), CanvasJS (http://canvasjs.com).

[16] para simplificar la búsqueda se realiza el filtro por Partido y no por domicilio ingresado

Consideraciones generales

- El prototipo debe ser desarrollado utilizando PHP, HTML5, CSS3 y MySQL, y **respetando el modelo en capas MVC**. Debe utilizar PDO como mecanismo de abstracción de bases de datos, teniendo la posibilidad de utilizar Doctrine como ORM. Debe tener en cuenta los conceptos de Semántica Web proporcionada por HTML5 siempre y cuando sea posible con una correcta utilización de las etiquetas del lenguaje.
- El trabajo será evaluado desde el servidor de la cátedra que cada grupo deberá gestionar mediante Git. **NO se aceptarán entregas que no estén realizadas en tiempo y forma en el servidor provisto por la cátedra**.
- El/la ayudante a cargo evaluará el progreso y la participación de cada integrante mediante las consultas presenciales y el seguimiento mediante GitLab.
- Toda vista (**HTML5** y **CSS3**) debe validar contra las especificaciones de la W3C (http://validator.w3.org/ y https://jigsaw.w3.org/css-validator/ respectivamente). En esta oportunidad puede utilizar **Bootstrap** u otro framework similar.
- **El uso de Twig como motor de plantillas es obligatorio**.
- No puede utilizar un framework PHP para el desarrollo.
- No puede utilizar un framework/generador de código para el desarrollo de la API de Instituciones.
- Para esta etapa, la entrega y aprobación son obligatorias. Todos y todas los/as integrantes deben presentarse a la defensa.
- El sistema no debe ser susceptible a SQL Injection, XSS ni CSRF.
- Tener en cuenta que el servidor de la cátedra utiliza los siguientes servicios y versiones:
    - Servidor Web: Apache 2.4.29
    - Servidor de Base de Datos: MariaDB 10.1.34
    - Intérprete PHP: Php 7.2

- **Importante**:
    - El proyecto podrá ser realizado de modo individual o en grupos de dos o tres integrantes (será responsabilidad de los y las estudiantes la conformación de los equipos de trabajo). Todos y todas los/as estudiantes cumplirán con la totalidad de la consigna, sin excepciones.
    - Aquellos trabajos que se verifique sean copias de otros, quedarán desaprobados para esa entrega, debiendo reentregar en caso de contar con dicha instancia o perderán la cursada si se trata de la última entrega.

Modelo de la Base de Datos

Para simplificar el desarrollo del trabajo ofrecemos el siguiente modelo de base de datos. Tener en cuenta que en el modelo también se encuentran las tablas (obra_social, tipo_documento, localidad, partido y region_sanitaria) de referencia que serán provistas por la **API de Referencias,** respecto de las tablas motivo_consulta y tipo_institucion, ya se encuentran cargadas con datos, no es necesario realizar el ABM.

Modelo Completo